REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE Technical Report	3. DATES COVERED (From - To) -

4. TITLE AND SUBTITLE Greedy Learning of Graphical Models with Small Girth	5a. CONTRACT NUMBER W911NF-11-1-0265
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 611102
6. AUTHORS Avik Ray, Sujay Sanghavi, Sanjay Shakkottai	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES AND ADDRESSES University of Texas at Austin 101 East 27th Street Suite 5.300 Austin, TX 78712 -1539	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 59441-NS.2

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

This paper presents three new greedy algorithms for learning discrete graphical models. The original greedy algorithm constructed the neighborhood of each node by sequentially adding nodes (or variable) in each step which currently produced the maximum decrease in its conditional entropy. Though simple, this did not always yield the correct graph when there are short cycles, since a non-neighbor may produce most decrease in the conditional entropy in a step and it gets added to its neighborhood.

15. SUBJECT TERMS
learnign graphical models, greedy methods

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Sanjay Shakkottai
UU	UU	UU	UU		19b. TELEPHONE NUMBER 512-471-5376

Report Title

Greedy Learning of Graphical Models with Small Girth

ABSTRACT

This paper presents three new greedy algorithms for learning discrete graphical models. The original greedy algorithm constructed the neighborhood of each node by sequentially adding nodes (or variable) in each step which currently produced the maximum decrease in its conditional entropy. Though simple, this did not always yield the correct graph when there are short cycles, since a non-neighbor may produce most decrease in the conditional entropy in a step and it gets added to its neighborhood.

The new algorithms can overcome this problem in three different ways. The recursive greedy algorithm iteratively runs the greedy algorithm in an inner loop, but each time only includes the last added node in the neighborhood set. On other hand the forward-backward greedy algorithm includes a node deletion step in each iteration, which prunes the incorrect nodes from the neighborhood set that may have been added earlier. Finally the greedy algorithm with pruning runs the greedy algorithm until completion and then removes all the incorrect neighbors. We give both analytical guarantees and empirical results for our algorithms. Running the algorithms with a candidate set of nodes instead of all the nodes and their greedy approach enables them to efficiently learn graphs even with small girth, which the previous greedy and convex optimization based algorithms cannot learn.

Greedy Learning of Graphical Models with Small Girth

Avik Ray, Sujay Sanghavi and Sanjay Shakkottai

***Abstract*—This paper presents three new greedy algorithms for learning discrete graphical models. The original greedy algorithm constructed the neighborhood of each node by sequentially adding nodes (or variable) in each step which currently produced the maximum decrease in its conditional entropy. Though simple, this did not always yield the correct graph when there are short cycles, since a non-neighbor may produce most decrease in the conditional entropy in a step and it gets added to its neighborhood.**

The new algorithms can overcome this problem in three different ways. The *recursive greedy algorithm* iteratively runs the greedy algorithm in an inner loop, but each time only includes the last added node in the neighborhood set. On other hand the *forward-backward greedy* algorithm includes a node deletion step in each iteration, which prunes the incorrect nodes from the neighborhood set that may have been added earlier. Finally the *greedy algorithm with pruning* runs the greedy algorithm until completion and then removes all the incorrect neighbors. We give both analytical guarantees and empirical results for our algorithms. Running the algorithms with a candidate set of nodes instead of all the nodes and their greedy approach enables them to efficiently learn graphs even with small girth, which the previous greedy and convex optimization based algorithms cannot learn.

I. Introduction

Graphical models have been widely used to tractably capture dependence relations amongst a collection of random variables in a variety of domains, ranging from statistical physics, social networks to biological applications [2]–[7]. A key challenge in these settings is in learning the precise dependence structure among the random variables – a problem that in the worst case is known to be NP hard in the number of variables [8]. However, with restrictions placed on the class of graphical models considered, it is known that polynomial time algorithms exist. Exploring the relationship between classes of graphical models that can be learnt, and the sample and computation complexity of doing so is an active field of research. One of the first results in this spirit is that by Chow and Liu [9], where efficient algorithms for learning tree-structured graphical models were developed. Since then, there have been several algorithms developed for learning restricted classes of graphical models (see Section I-B for more details).

A. Main Contributions

In this paper we propose three new greedy algorithms to find the Markov graph for any discrete graphical model. While greedy algorithms (that learn the structure by sequentially adding nodes and edges to the graph) tend to have low computational complexity, they are known to fail (i.e., do not determine the correct graph structure) in loopy graphs with low girth [13], even when they have access to exact statistics. This is because a non-neighbor can be the best node at a particular iteration; once added, it will always remain. Convex optimization based algorithms like in [10] by Ravikumar et al. (henceforth we call this the RWL algorithm) also cannot provide theoretical guarantees of learning in these situations. These methods require strong incoherence conditions to guarantee success. But such conditions may not be satisfied in even simple graphs with small girth [18]. **Example:** If we run the existing algorithms for an Ising model on a diamond network (Figure 2) with $D = 4$ the performance plot in Figure 1 shows that greedy and RWL algorithms fail to learn the correct graph even with large number of samples.

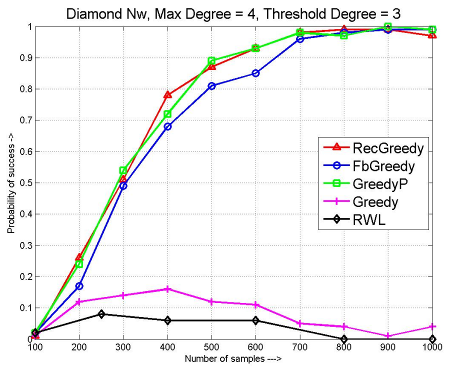

Fig. 1: Performance of different algorithms in an Ising model on diamond network with 6 nodes (Figure 2 with $D = 4$). Both the $Greedy(\epsilon)$ and RWL algorithms estimate an incorrect edge between nodes 0 and 5 therefore never recovers the true graph G, while our new $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$, $GreedyP(\epsilon)$ algorithms succeed.

In this paper, we present three algorithms that overcome this shortfall of greedy and convex optimization based algorithms.

- The *recursive greedy algorithm* is based on the observation that the *last* node added by the simple, naive greedy algorithm is always a neighbor; thus, we can use the naive greedy algorithm as an inner loop that, after every execution, yields just one more neighbor (instead of the entire set).
- The *forward-backward greedy algorithm* takes a different tack, interleaving node addition (forward steps) with node removal (backward steps). In particular, in every iteration, the algorithm looks for nodes in the existing set that have a very small marginal effect; these are removed. Note that these nodes may have had a big effect in a previous

A. Ray, S. Sanghavi and S. Shakkottai are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, USA, Emails: `avik@utexas.edu`, `sanghavi@mail.utexas.edu`, `shakkott@austin.utexas.edu`. An earlier version of this work has appeared in the Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing, 2012 [1]. We would like to acknowledge NSF grants 0954059 and 1017525, and ARO grant W911NF-11-1-0265.

iteration when they were added, but the inclusion of subsequent nodes shows them to not have enough of a direct effect.

- Our third algorithm, namely the *greedy algorithm with pruning*, first runs the greedy algorithm until it is unable to add any more nodes to the neighborhood estimate. Subsequently, it executes a node pruning step that identifies and removes all the incorrect neighbors that were possibly included in the neighborhood estimate by the greedy algorithm.
- For all these algorithms, we show that a simple node selection step at the beginning followed by any one of the recursive, forward-backward or greedy with pruning algorithms can efficiently learn the structure of a large class of graphical models even when they have small cycles. We calculate the sample complexity and computational (number of iterations) complexity for these algorithms (with high probability) under non-degeneracy and correlation decay assumptions (Theorems 3 and 4);
- Finally we present numerical results that indicate tractable sample and computational complexity for loopy graphs (diamond graph, grid).

B. Related Work

Several approaches have been taken so far to learn the graph structure of MRF in presence of cycles. These can be broadly divided into three classes – search based, convex-optimization based, and greedy methods.

Search based algorithms like local independence test (LIT) by Bresler et al. in [11] and the conditional variation distance thresholding (CVDT) by Anandkumar et al. in [14] try to find the smallest set of nodes through exhaustive search, conditioned on which either a given node is independent of rest of the nodes in the graph, or a pair of nodes are independent of each other. These algorithms have a fairly good sample complexity, but due to exhaustive search they have a high computation complexity. Also to run these algorithms one needs to know some additional information about the graph structure. For example the local independence test requires the knowledge of the maximum degree of the graph and the CVDT algorithm requires the knowledge of the maximum size of the local separator for the graph.

In case of Ising models a convex optimization based learning algorithm was proposed in [10] by Ravikumar et al. This was further generalized for any pairwise graphical model in [12]. These algorithms try to construct a pseudo likelihood function using the parametric form of the distribution such that it is convex and try to maximize it over the parameter values. The optimized parameter values in effect reveal the Markov graph structure. These algorithms have a very good sample complexity of $\Omega(\Delta^3 \log p)$, where Δ is the maximum degree of a node and p is the total number of nodes. However these algorithms require a strong incoherence assumption to guarantee its success. In [18] Bento et al. showed that even for a large class of Ising models the incoherence conditions are not satisfied hence the convex optimization based algorithms fail. They also show that in Ising models with weak long range correlation, a simple low complexity thresholding algorithm can correctly learn the graph.

Recently a greedy learning algorithm was proposed in [13] which tries to find the minimum value of the conditional entropy of a particular node in order to estimate its neighborhood. We call this algorithm as $Greedy(\epsilon)$. It is an extension of the Chow-Liu algorithm to graphs with cycles. It was shown that for graphs with correlation decay and large girth this exactly recovers the graph G. However it fails for graphs with small cycles. A forward-backward greedy algorithm based on convex optimization was also presented recently by Jalali et al. in [19], which works for any pairwise graphical model. This required milder assumption than in [10] and also gives a better sample complexity.

This paper is organized as follows. First we review the definition of a graphical model and the graphical model learning problem in section II. The three greedy algorithms are described in section III. Next we give sufficient conditions for the success of the greedy algorithms in section IV. In section V we present the main theorems showing the performance of the recursive greedy, forward-backward and greedy with pruning algorithms. We compare the performance of our algorithm with other well known algorithms in section VI. In section VII we present some simulation results. The proofs are presented in the appendix.

II. Brief Review: Graphical Models

In this section we briefly review the general graphical model and the Ising model. Let $X = (X_1, X_2, \ldots, X_p)$ be a random vector over a discrete set $\mathcal{X}^p$, where $\mathcal{X} = \{1, 2, \ldots, m\}$. $X_S = (X_i : i \in S)$ denote the random vector over the subset $S \subseteq \{1, 2, \ldots, p\}$. Let $G = (V, E)$ denote a graph having p nodes. Let Δ be the maximum degree of the graph G and Δ_i be the degree of the i^{th} node. An undirected graphical model or Markov random field is a tuple $M = (G, X)$ such that each node in G corresponds to a particular random variable in X. Moreover G captures the Markov dependence between the variables X_i such that absence of an edge (i, j) implies the conditional independence of variables X_i and X_j given all the other variables.

For any node $r \in V$, let $\mathcal{N}_r$ denote the set of neighbors of r in G. Then the distribution $\mathbb{P}(X)$ has the special Markov property that for any node r, X_r is conditionally independent of $X_{V\setminus\{r\}\bigcup\mathcal{N}_r}$ given $X_{\mathcal{N}_r} = \{X_i : i \in \mathcal{N}_r\}$, the neighborhood of r, i.e.

$$\mathbb{P}(X_r | X_{V\setminus r}) = \mathbb{P}(X_r | X_{\mathcal{N}_r}) \tag{1}$$

Ising Model: An Ising model is a pairwise graphical model where X_i take values in the set $\mathcal{X} = \{-1, 1\}$. For this paper we also consider the node potentials as zero (the zero field Ising model). Hence the distribution take the following simplified form.

$$\mathbb{P}_\Theta(X = x) = \frac{1}{Z} \exp\left\{ \sum_{(i,j)\in E} \theta_{ij} x_i x_j \right\} \tag{2}$$

Algorithm	Graph family	Sample complexity	Computation complexity
Bresler et al.	Δ degree limited	$\Omega\left(\lvert\mathcal{X}\rvert^{4\Delta}\Delta\log p\right)$	$O\left(p^{2\Delta+1}\log p\right)$
CVDT	$(\Delta,\gamma)-$ local separation	$\Omega\left(\lvert\mathcal{X}\rvert^{2\Delta}(\Delta+2)\log p\right)$	$O\left(p^{\Delta+2}\right)$
RWL	Δ degree limited, Ising model, incoherence	$\Omega\left(\Delta^3\log p\right)$	$O\left(p^4\right)$
Jalali et al.	Δ degree limited, pairwise graphical model, RSC	$\Omega\left(\Delta^2\log p\right)$	$O\left(p^4\right)$
Greedy	Δ degree limited, large girth, correlation decay	$\Omega\left(\lvert\mathcal{X}\rvert^{4\Delta}\log p\right)$	$O\left(p^2\Delta\right)$
RecGreedy	Δ degree limited, correlation decay	$\Omega\left(\frac{\lvert\mathcal{X}\rvert^{2\log\lvert\mathcal{X}\rvert/\epsilon}}{\epsilon^5}\log p\right)$	$O\left(p^2+p\Delta\frac{\xi}{\epsilon}\right)$
FbGreedy	Δ degree limited, correlation decay	$\Omega\left(\frac{\lvert\mathcal{X}\rvert^{4\log\lvert\mathcal{X}\rvert/((1-\alpha)\epsilon)}}{\epsilon^5(1-\alpha)\alpha^4}\log p\right)$	$O\left(p^2+p\frac{\xi}{(1-\alpha)\epsilon}\right)$
GreedyP	Δ degree limited, correlation decay	$\Omega\left(\frac{\lvert\mathcal{X}\rvert^{2\log\lvert\mathcal{X}\rvert/\epsilon}}{\epsilon^5}\log p\right)$	$O\left(p^2+p\frac{(\xi+1)}{\epsilon}\right)$
Bento et al.	Δ degree limited, Ising model, correlation decay	$\Omega\left(\frac{\Delta^2}{(1-2\Delta\tanh\theta)^2}\log p\right)$	$O\left(p^2\right)$

TABLE I: Performance comparison between different discrete graphical model learning algorithms in literature. Δ denotes the maximum degree of the graph and p is the number of nodes. $\mathcal{X}$ is the alphabet set from which a random variable take its value in the discrete graphical model. ξ represents the super-neighborhood size (described in Section IV-C). ϵ is a non-degeneracy parameter (see Section IV). α is an input parameter to determine the elimination threshold in $FbGreedy$ algorithm (see Section III-B). θ is the edge weight parameter of the Ising model in [18].

where $x_i, x_j \in \{-1, 1\}$, $\theta_{ij} \in \mathbb{R}$ and Z is the normalizing constant.

Graphical Model Selection: The graphical model selection problem is as follows. Given n independent samples $\mathcal{S}_n = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ from the distribution $\mathbb{P}(X)$, where each $x^{(i)}$ is a p dimensional vector $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}) \in \{1, \ldots, m\}^p$, the problem is to estimate the Markov graph G corresponding to the distribution $\mathbb{P}(X)$ by recovering the correct edge set E. This problem is NP hard in general and has been solved only under special assumptions on the graphical model structure. In some cases a learning algorithm is able to find the correct neighborhood of each node $v \in V$ with a high probability and hence recover the true topology of the graph. Table I has a brief survey of the most relevant methods for discrete graphical models.

We describe some notations. For a subset $S \subseteq V$, we define $P(x_S) = \mathbb{P}(X_S = x_S)$, $x_S \in \mathcal{X}^{|S|}$. The empirical distribution $\widehat{P}(X)$ is the distribution of X computed from the samples. Let $i \in V - S$, the entropy of the random variable X_i conditioned on X_S is written as $H(X_i|X_S)$. The empirical entropy calculated corresponding to the empirical distribution $\widehat{P}$ is denoted by $\widehat{H}$. If P and Q are two probability measures over a finite set $\mathcal{Y}$, then the total variational distance between them is given by, $||P - Q||_{TV} = \frac{1}{2}\sum_{y\in\mathcal{Y}} |P(y) - Q(y)|$.

III. Greedy Algorithms

In this section we describe three new greedy algorithms for learning the structure of a MRF.

Main idea: The algorithms can be divided into two steps. The first step common to all algorithms is a node pruning step called *super-neighborhood selection* (described in detail in Section IV). This step generates a collection of super-neighborhoods $\mathcal{S} = \{S_i \subseteq V : \mathcal{N}_i \subseteq S_i \text{ and } i \notin S_i \ \forall i \in V\}$ one for each $i \in V$, such that $|S_i|$ is small. This super-neighborhood set $\mathcal{S}$ is the input to the second step of the algorithms which is described next.

A. Recursive Greedy Algorithm

Idea: Consider first the simpler setting when we have infinite samples (i.e. access to the true population quantity). The simple naive greedy algorithm [13] adds nodes to the neighborhood stopping when no further strict reduction in conditional entropy is possible. This stopping will happen when the true neighborhood is a subset of the estimated neighborhood. Our key observation here is that the *last* node to be added by the naive greedy algorithm will always be in the true neighborhood, since inclusion of the last neighbor in the conditioning set enables it to reach the minimum conditional entropy. We leverage this observation by using the naive greedy algorithm as an inner loop; at the end of every run of this inner loop, we pick only the last node and add it to the estimated neighborhood. The next inner loop starts with this added node as an initial condition, and finds the next one. Hence the algorithm discovers a neighbor in each run of the innermost loop and finds all the neighbors of a given node i in exactly Δ_i iterations of the outer loop.

The above idea works as long as every neighbor has a measurable effect on the conditional entropy, even when there are several other variables in the conditioning. The algorithm is $RecGreedy(\epsilon)$, pseudocode detailed below. It needs a non-degeneracy parameter ϵ, which is the threshold for how much effect each neighbor has on the conditional entropy.

B. Forward-Backward Greedy Algorithm

Our second algorithm takes a different approach to fix the problem of spurious nodes added by the naive greedy algorithm, by adding a backward step at every iteration that prunes nodes it detects as being spurious. In particular, after every forward step that adds a node to the estimated neighborhood, the algorithm finds the node in this new estimated neighborhood that has the smallest individual effect on the new conditional entropy. If this is too small, this node is removed from the estimated neighborhood.

The algorithm, $FbGreedy(\epsilon, \alpha)$, is given in pseudocode. It takes two input parameters beside the samples. The first is the same non-degeneracy parameter ϵ as in the $RecGreedy(\epsilon)$ algorithm. The second parameter $\alpha \in (0, 1)$ is utilized by

Algorithm 1 $RecGreedy(\epsilon)$

```
1: Generate super-neighborhood $\mathcal{S}$
2: for $i = 1$ to $|V|$ do
3:     $\widehat{N}(i) \leftarrow \phi$
4:     iterate ← TRUE
5:     while iterate do
6:         $\widehat{T}(i) \leftarrow \widehat{N}(i)$
7:         last ← 0
8:         complete ← FALSE
9:         while ! complete do
10:            $j = \arg\min_{k \in S_i \setminus \widehat{T}(i)} \widehat{H}(X_i | X_{\widehat{T}(i)}, X_k)$
11:            if $\widehat{H}(X_i | X_{\widehat{T}(i)}, X_j) < \widehat{H}(X_i | X_{\widehat{T}(i)}) - \frac{\epsilon}{2}$ then
12:                $\widehat{T}(i) \leftarrow \widehat{T}(i) \bigcup \{j\}$
13:                last ← $j$
14:            else
15:                if last ! = 0 then
16:                    $\widehat{N}(i) \leftarrow \widehat{N}(i) \bigcup \{\text{last}\}$
17:                else
18:                    iterate ← FALSE
19:                end if
20:                complete ← TRUE
21:            end if
22:        end while
23:    end while
24: end for
```

the algorithm to determine the threshold of elimination in the backward step. We will see later that this parameter also helps to trade-off between the sample and computation complexity of the $FbGreedy(\epsilon, \alpha)$ algorithm. The algorithm stops when there are no further forward or backward steps.

C. Greedy Algorithm with Pruning

The third algorithm overcomes the problem of non-neighbor inclusion in the $Greedy(\epsilon)$ algorithm by adding a node pruning step after the execution of the greedy algorithm (similar to the backward step in $FbGreedy(\epsilon, \alpha)$). In this algorithm, after running the $Greedy(\epsilon)$ algorithm, the pruning step declares a neighbor node to be spurious if its removal from the neighborhood estimate does not significantly increase the final conditional entropy. These spurious nodes are removed to result in an updated neighborhood estimate for each node. In the original $Greedy(\epsilon)$ algorithm this step was impractical since the size of the estimated neighborhood could become very large [13] in a general graph. However, our pre-processing step (super-neighborhood selection, see Section IV) effectively precludes this possibility and leads to an effcient and correct algorithm.

The pseudocode of this greedy algorithm with node-pruning – $GreedyP(\epsilon)$ – is given in Algorithm 3. In addition to the samples, the input is again a non-degeneracy parameter ϵ similar to $RecGreedy(\epsilon)$ and $FbGreedy(\epsilon, \alpha)$ algorithms.

Algorithm 2 $FbGreedy(\epsilon, \alpha)$

```
1: Generate super-neighborhood $\mathcal{S}$
2: for $i = 1$ to $|V|$ do
3:     $\widehat{N}(i) \leftarrow \phi$
4:     added ← FALSE
5:     complete ← FALSE
6:     while ! complete do                                  ▷ Forward Step:
7:         $j = \arg\min_{k \in S_i \setminus \widehat{N}(i)} \widehat{H}(X_i | X_{\widehat{N}(i)}, X_k)$
8:         if $\widehat{H}(X_i | X_{\widehat{N}(i)}, X_j) < \widehat{H}(X_i | X_{\widehat{N}(i)}) - \frac{\epsilon}{2}$ then
9:             $\widehat{N}(i) \leftarrow \widehat{N}(i) \bigcup \{j\}$
10:            added ← TRUE
11:        else
12:            added ← FALSE
13:        end if                                           ▷ Backward Step:
14:        $l = \arg\min_{k \in \widehat{N}(i)} \widehat{H}(X_i | X_{\widehat{N}(i) \setminus k})$
15:        if $\widehat{H}(X_i | X_{\widehat{N}(i) \setminus l}) - \widehat{H}(X_i | X_{\widehat{N}(i)}) < \frac{\alpha\epsilon}{2}$ then
16:            $\widehat{N}(i) \leftarrow \widehat{N}(i) \setminus \{l\}$
17:        else
18:            if ! added then
19:                complete ← TRUE
20:            end if
21:        end if
22:    end while
23: end for
```

Algorithm 3 $GreedyP(\epsilon)$

```
1: Generate super-neighborhood $\mathcal{S}$
2: for $i = 1$ to $|V|$ do
3:     Run $Greedy(\epsilon)$ within set $S_i$
4:     for $j \in \widehat{N}(i)$ do
5:         if $\widehat{H}(X_i | X_{\widehat{N}(i) \setminus j}) - \widehat{H}(X_i | X_{\widehat{N}(i)}) < \frac{\epsilon}{2}$ then
6:             $\widehat{N}(i) \leftarrow \widehat{N}(i) \setminus j$
7:         end if
8:     end for
9: end for
```

IV. Sufficient Conditions for Markov Graph Recovery

In this section we describe the sufficient conditions which guarantees that the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms recover the correct Markov graph G.

A. Non-degeneracy

Our non-degeneracy assumption requires every neighbor have a significant enough effect. Other graphical model learning algorithms require similar assumptions to ensure correctness [10], [11], [13].

(A1) Non-degeneracy condition: Consider the graphical model $M = (G, X)$, where $G = (V, E)$. Then for all $i \in V$ and $A \subset V$ such that $\mathcal{N}_i \not\subset A$ the following condition holds. Let $j \in \mathcal{N}_i$ and $j \notin A$. Then there exists $\epsilon > 0$ such that

$$H(X_i | X_A, X_j) \quad < \quad H(X_i | X_A) - \epsilon \tag{3}$$

Thus by adding a neighboring node to any conditioning set that does not already contain it, the conditional entropy strictly decreases by at least ϵ. Also the above condition together with the local Markov property (1) implies that the conditional entropy attains a unique minimum at $H(X_i|X_{\mathcal{N}_i})$.

B. Correlation Decay

Correlation decay broadly means that the influence of a random variable on the distribution of another gradually decreases as the path distance between the corresponding nodes increase in the graph G. In [18] Bento et al. showed that learning graphical models become more difficult in absence of some sort of correlation decay. Many different forms of correlation decay have been assumed in MRF learning algorithms [11], [13], [14]. We assume a weak form of correlation decay similar to the weak spatial mixing assumption in [20]. First we define the following quantity.

Definition 1 *Consider the graphical model $M = (G, X)$. Let $i, j \in V$. Define $\phi_i(j) = \max_{x \neq x'} ||P(X_i|X_j = x) - P(X_i|X_j = x')||_{TV}$. The corresponding function calculated from the empirical distribution $\widehat{P}$ is denoted as $\widehat{\phi}_i(j)$.*

$\phi_i(j)$ denotes the maximum variation distance between the conditional distribution of X_i conditioned on two different values of X_j. Now the correlation decay assumption is the following.

(A2) Correlation decay: For the graphical model $M = (G, X)$ there exists a monotonic decreasing function $f : \mathbb{Z} \to \mathbb{R}$ such that for any $i, j \in V$

$$\phi_i(j) < f(d(i,j)) \tag{4}$$

where $d(i,j)$ is the graph distance between nodes i and j. It can be shown that the correlation decay assumption in [13] implies (4) for any monotonic decreasing function $f(.)$ (scaled appropriately). Hence this is a weaker assumption. Next we give an example when the decay function $f(.)$ is exponential.

Example 1 (*Exponential correlation decay*)

It can be shown that in any graphical model $M = (G, X)$ if Dobrushin's condition holds then M exhibits an exponential correlation decay. First we restate the definition of influence coefficient from [15], [16].

Definition 2 *Influence coefficient: For any $i, j \in V$ the influence coefficient of node j on node i is*

$$C_{ij} = \max_{\substack{y,z \in \mathcal{X}^{|V|-1} \\ y_k = z_k \ \forall k \neq j}} ||P(X_i|X_{V\setminus i} = y) - P(X_i|X_{V\setminus i} = z)||_{TV}$$

Note that due to the Markov property of the graph $C_{ij} = 0$ for all $j \notin \mathcal{N}_i$. Dobrushin's condition [17], [21] is the following.

Dobrushin's condition: Let C_{ij} be the influence coefficient of node j on node i. Then Dobrushin's condition require

$$\gamma = \sup_{i \in V} \left(\sum_{j \in V} C_{ij} \right) < 1 \tag{5}$$

In an Ising model (2) with maximum degree Δ and $\theta_{ij} = \theta$ the Dobrushin's condition corresponds to $\gamma = \Delta \tanh 2\theta < 1$ [21]. The following lemma connects this to assumption (A2).

Lemma 1 ([21]) *Suppose Dobrushin's condition holds for a Markov random field. Then,*

$$\phi_i(j) \leq \frac{\gamma^{d(i,j)}}{1-\gamma}$$

where γ is given by (5).

Hence in this case $f(x) = \frac{\gamma^x}{1-\gamma}$ is an exponentially decaying function.

C. Super-Neighborhood Selection

In this section we describe a method to choose a super-neighborhood S_i for each node $i \in V$ in the first step of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms when there is correlation decay. Recall that a super-neighborhood set S_i for any node i is a subset of nodes which contain the true neighborhood $\mathcal{N}_i$.

Before we describe the procedure, we motivate the need for a super-neighborhood selection method. First, observe that if we run Algorithm 1, 2, and 3 with $S_i = V$ for all $i \in V$ and with exact distribution $P(X)$ known, under the non-degeneracy assumption (A1) the algorithm correctly outputs the true neighborhood $\mathcal{N}_i$ with a proper ϵ. However the problem is that for an arbitrary graphical model (or any graphical model with the super-neighborhood set to be very large), the size of the conditioning set $\widehat{T}(i)$ in $RecGreedy(\epsilon)$ or $\widehat{N}(i)$ in $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ can also become very large. This implies that the number of samples required to get a good estimate of the conditional entropy $H(X_i|X_A)$ is $\Omega(|\mathcal{X}|^{|A|+1})$ will be exponentially large (a good estimate is needed to ensure Algorithm 1, 2 and 3 give the correct graph G with a high probability). In order to mitigate this problem we need to appropriately bound the size of the set $\widehat{T}(i)$, $\widehat{N}(i)$. To do this we choose a super-neighborhood S_i such that $\sup_{i \in V} |S_i| := \xi$ and $\xi = O(\log p)$. Then the size of the conditioning set never exceeds ξ and the number of samples needed by the algorithm (sample complexity) will be at most polynomial in p.

The problem of super-neighborhood selection becomes easier under the correlation decay assumption (A2). Let β be such that,

$$\min_{i \in V, j \in \mathcal{N}_i} \phi_i(j) = \beta \tag{6}$$

The super-neighborhood is then selected as follows.

$$S_i = \{j \in V | \widehat{\phi}_i(j) \geq \frac{\beta}{2}\} \tag{7}$$

Remark: Note that there may be other ways to generate a super-neighborhood based on domain knowledge/structural properties of the system (e.g. in social networks, weather forecasting). All that is needed for our algorithms is to have a super-neighborhood of small size.

Definition 3 (Super-neighborhood radius) *The super-neighborhood radius R is defined as*

$$R = \min\{x \in \mathbb{Z} | f(x) < \beta/2\} \tag{8}$$

We assume that R exists and R does not grow with p. i.e., $R = O(1)$.

Now with correlation decay (A2) this super-neighborhood selection procedure (7) is successful with a high probability with only $\Omega(\log p)$ samples, where by success we mean S_i will contain the true neighborhood $\mathcal{N}_i$ and $\xi = \max_{i \in V} |S_i|$ is small. This is shown by the following lemmas. We define the minimum marginal probability P_{min} as $P_{min} = \min_{i \in V, x_i \in \mathcal{X}} P(X_i = x_i)$.

Lemma 2 *Consider a graphical model $M = (G, X)$ with distribution $P(X)$, $X \in \mathcal{X}^p$. Let $0 < \delta_1 < 1$. Then if the number of i.i.d. samples*

$$n > \frac{32|\mathcal{X}|^4}{\beta^2 P_{min}^2}\left[2 \log |\mathcal{X}| p + \log \frac{2}{\delta_1}\right]$$

we have with probability at least $1 - \delta_1$

$$|P(X_i|X_j) - \widehat{P}(X_i|X_j)| < \frac{\beta}{4|\mathcal{X}|} \tag{9}$$

for all $i, j \in V$, where β is given by (6).

Lemma 3 *Let a graphical model $M = (G, X)$ satisfy assumption (A2). Let $0 < \delta_1 < 1$. Then with probability greater than $1 - \delta_1$, $\mathcal{N}_i \subseteq S_i$ for all $i \in V$ when the number of i.i.d. samples $n = \Omega(\log \frac{p}{\delta_1})$.*

Lemma 4 *Consider a graphical model $M = (G, X)$ with maximum degree Δ satisfying assumption (A2) with decay function $f(.)$. Let $0 < \delta_2 < 1$. Then with probability greater than $1 - \delta_2$ we have $|S_i| < \Delta_i \Delta^{R-1}$ when the number of samples $n = \Omega(\log \frac{p}{\delta_2})$, where R is given by (8).*

Lemmas 2, 3, 4 follow from Azuma's concentration inequality as presented in Appendix. Now we try to characterize the maximum super-neighborhood size ξ in bounded degree graphs with exponential correlation decay and in Ising models.

Theorem 1 *Consider a graphical model $M = (G, X)$ with maximum degree Δ satisfying Dobrushin's condition (5). Let $0 < \delta_4 < 1$. Then with probability greater than $1 - \delta_4$ we have $\xi < \Delta^{\log \frac{2}{(1-\gamma)\beta} / \log \frac{1}{\gamma}}$ when the number of samples $n = \Omega(\log \frac{p}{\delta_4})$, where β is given by equation (6).*

Theorem 2 *Consider a zero field Ising model with equal edge weights θ, distribution given by equation (2). Let maximum degree Δ satisfy $\Delta \tanh 2\theta < 1$. Let $0 < \delta_5 < 1$. When the number of samples $n = \Omega(\log \frac{p}{\delta_5})$, with probability greater than $1 - \delta_5$ we have $\xi < \Delta^{\log((1-\Delta \tanh 2\theta) \tanh \theta/2) / \log(\Delta \tanh 2\theta)}$.*

Theorems 1 and 2 mainly follow from Lemma 1 and 4. Detailed proof is given in Appendix. A super-neighborhood selection process for graphs with exponential correlation decay is available in [11]. However the super-neighborhood selection in our setting applies to graphs exhibiting a weaker form of correlation decay. Further, unlike in [11] where the main purpose for super-neighborhood selection was to reduce the computational complexity of the search based algorithm, in our case super-neighborhood selection reduces both the sample and computational complexity of the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms.

V. Main Result

In this section we state our main result showing the performance of the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms. First we state some useful lemmas. We restate the first lemma from [13], [23] that will be used to show the concentration of the empirical entropy $\widehat{H}$ with samples.

Lemma 5 *Let P and Q be two discrete distributions over a finite set $\mathcal{X}$ such that $||P - Q||_{TV} \leq \frac{1}{4}$. Then,*

$$|H(P) - H(Q)| \leq 2||P - Q||_{TV} \log \frac{|\mathcal{X}|}{2||P - Q||_{TV}}$$

The following two lemmas bound the number of steps in $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms which also guarantees their convergence.

Lemma 6 *The number of greedy steps in each recursion of the $RecGreedy(\epsilon)$ and in $GreedyP(\epsilon)$ algorithm is less than $\frac{2 \log |\mathcal{X}|}{\epsilon}$.*

Proof: In each step the conditional entropy is reduced by an amount at least $\epsilon/2$. Since the maximum reduction in entropy possible is $\widehat{H}(X_i) \leq \log |\mathcal{X}|$, the number of steps is upper bounded by $\frac{2 \log |\mathcal{X}|}{\epsilon}$.

■

Note that the $GreedyP(\epsilon)$ algorithm will take at least Δ steps to include all the neighbors in the conditioning set. Hence $\frac{2 \log |\mathcal{X}|}{\epsilon} \geq \Delta$.

Lemma 7 *The number of steps in the $FbGreedy(\epsilon, \alpha)$ is upper bounded by $\frac{4 \log |\mathcal{X}|}{\epsilon(1-\alpha)}$.*

Proof: Note that as long as the forward step is active (which occurs till all neighbors are included in the conditioning set $\widehat{N}(i)$), in each step the conditional entropy reduces by at least $(1 - \alpha)\epsilon/2$. Hence all the neighbors are included within $\frac{2 \log |\mathcal{X}|}{(1-\alpha)\epsilon}$ steps. The number of non-neighbors included in the conditioning set is also bounded by $\frac{2 \log |\mathcal{X}|}{(1-\alpha)\epsilon}$. Thus it will take at most the same number backward steps to remove the non-neighbors. Hence the total number of steps is at most $\frac{4 \log |\mathcal{X}|}{(1-\alpha)\epsilon}$.

■

We now state our main theorem showing the performance of Algorithms 1, 2 and 3.

Theorem 3 *Consider a MRF over a graph G with maximum degree Δ, having a distribution $P(X)$.*
*1) **Correctness (non-random):** Suppose (A1) holds and the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms have access to the true conditional entropies therein, then they correctly estimate the graph G.*
*2) **Sample complexity:** Suppose (A1) holds, super-neighborhoods S_i are given and super-neighborhood size $|S_i| < \xi$, for all $i \in V$. Let $0 < \delta < 1$.*

- *When the number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$ the $RecGreedy(\epsilon)$ correctly estimates G with probability greater than $1-\delta$.*
- *When the number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{4\log|\mathcal{X}|/((1-\alpha)\epsilon)}}{\epsilon^5(1-\alpha)\alpha^4}\log\frac{p}{\delta}\right)$ the $FbGreedy(\epsilon, \alpha)$ correctly estimates G with probability greater than $1-\delta$, for $0 < \alpha < 1$.*
- *When the number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$ the $GreedyP(\epsilon)$ correctly estimates G with probability greater than $1-\delta$.*

The proof of correctness with true conditional entropies known is straightforward under non-degenerate assumption (A1). The proof in presence of samples is based on Lemma 8 similar to Lemma 2 in [13] showing the concentration of empirical conditional entropy, which is critical for the success of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms. We show that when super-neighborhoods S_i are given and $|S_i| \leq \xi$ for all $i \in V$, with sufficiently many samples the empirical distributions and hence the empirical conditional entropies also concentrate around their true values with a high probability. This will ensure that algorithms 1, 2 and 3 correctly recover the Markov graph G. The complete proof is presented in Appendix.

Lemma 8 *Consider a graphical model $M = (G, X)$ with distribution $P(X)$. Let $0 < \delta_3 < 1$. If the number of samples*

$$n > \frac{8|\mathcal{X}|^{2(s+2)}}{\zeta^4}\left[(s+1)\log 2p|\mathcal{X}| + \log\frac{1}{\delta_3}\right]$$

then with probability at least $1-\delta_3$

$$|\widehat{H}(X_i|X_S) - H(X_i|X_S)| < \zeta$$

for any $S \subset V$ such that $|S| \leq s$.

Lemma 8 follows from Lemma 5 and Azuma's inequality. Although the sample complexities of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms are slightly more than other non-greedy algorithms [10], [11], [14], the main appeal of these greedy algorithms lie in their low computation complexity. The following theorem characterizes the computation complexity of Algorithms 1, 2 and 3. When calculating the run-time, each arithmetic operation and comparison is counted as an unit-time operation. For example to execute line 10 in Algorithm 1, each comparison takes an unit-time and each entropy calculation takes $O(n)$ time (since there are n samples using which the empirical conditional entropy is calculated). Since there are at most $|S_i| \leq \xi$ comparisons the total time required to execute this line is $O(n\xi)$.

Theorem 4 (Run-time) *Consider a graphical model $M = (G, X)$, with maximum degree Δ, satisfying assumptions (A1) and $|S_i| < \xi$, for all $i \in V$. Then the second step has an expected run-time of,*

- *$O(\delta p\xi^3 n + (1-\delta)\frac{p}{\epsilon}\Delta\xi n)$ for the $RecGreedy(\epsilon)$ algorithm.*
- *$O(\frac{p}{(1-\alpha)\epsilon}\xi n)$ for the $FbGreedy(\epsilon, \alpha)$ algorithm.*
- *$O(\delta p\xi(\xi+1)n + (1-\delta)\frac{p}{\epsilon}(\xi+1)n)$ for the $GreedyP(\epsilon)$ algorithm.*

The proofs of Theorem 4 are given in Appendix.
Remark: Suppose that $\frac{4\log|\mathcal{X}|}{\epsilon(1-\alpha)} < \xi$. Then if we take $\alpha < \frac{\Delta-1}{\Delta}$ the $FbGreedy(\epsilon, \alpha)$ has a better run time guarantee than the $RecGreedy(\epsilon)$ algorithm for small δ. But when $\Delta\xi < \frac{2\log|\mathcal{X}|}{\epsilon(1-\alpha)}$ then the $RecGreedy(\epsilon)$ algorithm has a better guarantee. Also when $\alpha < \frac{1}{\xi+1}$, $FbGreedy(\epsilon, \alpha)$ has a better runtime guarantee than $GreedyP(\epsilon)$ algorithm. Note that the super-neighborhood selection step in Equation (7) has an additional complexity of $O(p^2)$.

VI. Performance Comparison

In this section we compare the performance of the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms with other graphical model learning algorithms.

A. Comparison with $Greedy(\epsilon)$ algorithm:

The $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms are strictly better than the $Greedy(\epsilon)$ algorithm in [13]. This is because Algorithms 1, 2 and 3 always find the correct graph G when the $Greedy(\epsilon)$ finds the correct graph, but they are applicable to a wider class of graphical models since they do not require the assumption of large girth to guarantee its success. Further the correlation decay assumption (A2) in this paper is weaker than the assumption in [13]. Note that $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms use the $Greedy(\epsilon)$ algorithm as an intermediate step. Hence when $Greedy(\epsilon)$ finds the true neighborhood $\mathcal{N}_i$ of node i, $RecGreedy(\epsilon)$ algorithm will find the correct neighborhood in each of the recursive steps and $FbGreedy(\epsilon, \alpha)$, $GreedyP(\epsilon)$ algorithms output the correct neighborhood directly without having to utilize any of the backward steps or the pruning step respectively. Hence $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms also succeed in finding the true graph G. We now demonstrate a clear example of a graph where $Greedy(\epsilon)$ fails to recover the true graph but the Algorithms 1, 2 and 3 are successful. This example is also presented in [13]. Consider an Ising model on the graph in Figure 2. We have the following proposition.

Proposition 1 *Consider an Ising model with $V = \{0, 1, \ldots, D, D+1\}$ and $E = \{(0,i), (i, D+1)\ \forall i\ : 1 \leq i \leq D\}$ with a distribution function $P(x) = \frac{1}{Z}\prod_{(ij)\in E} e^{\theta x_i x_j}$, $X_i \in \{1, -1\}$. Then with $D > \frac{2\theta}{\log\cosh(2\theta)} + 1$ we have*

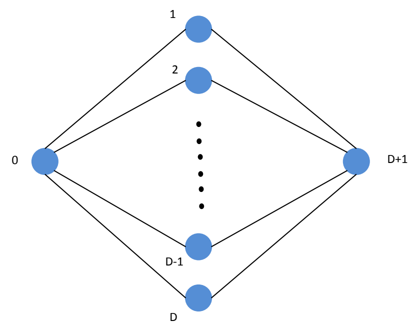

Fig. 2: An example of a diamond network with $D + 2$ nodes and maximum degree D where $Greedy(\epsilon)$ fails but $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms correctly recover the true graph.

$$H(X_0|X_{D+1}) < H(X_0|X_1)$$

The proof follows from straightforward calculation (see Appendix). Hence for the Ising model considered above (Figure 2) with $D > \frac{2\theta}{\log\cosh(2\theta)} + 1$ the $Greedy(\epsilon)$ incorrectly includes node $D+1$ in the neighborhood set in the first step. However with an appropriate ϵ the MRF satisfies assumption $(A1)$. Hence by taking $S_i = V$, Theorem 3 ensures that the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms correctly estimate the graph G.

B. Comparison with search based algorithms:

Search based graphical model learning algorithms like the Local Independence Test (LIT) by Bresler et al. [11] and the Conditional Variation Distance Thresholding (CVDT) by Anandkumar et al. [14] generally have good sample complexity, but high computation complexity. As we will see the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms have slightly more sample complexity but significantly lower computational complexity than the search based algorithms. Moreover to run the search based algorithms one needs to know the maximum degree Δ for LIT and the maximum size of the separator η for the CVDT algorithm. However the greedy algorithms can be run without knowing the maximum degree of the graph.

For bounded degree graphs the LIT algorithm has a sample complexity of $\Omega(|\mathcal{X}|^{4\Delta}\Delta\log\frac{2p}{\delta})$. Without any assumption on the maximum size of the separator, for bounded degree graphs the CVDT algorithm also has a similar sample complexity of $\Omega(|\mathcal{X}|^{2\Delta}(\Delta+2)\log\frac{p}{\delta})$. Note that the quantity P_{min} in the sample complexity expression for CVDT algorithm (Theorem 2 in [14]) is the minimum probability of $P(X_S = x_S)$ where $|S| \leq \eta+1$. This scales with Δ as $P_{min} \leq \frac{1}{|\mathcal{X}|^{\eta+1}}$. For general degree bounded graphs we have $\eta = \Delta$. The sample complexity for $RecGreedy(\epsilon)$, $GreedyP(\epsilon)$ and $FbGreedy(\epsilon, \alpha)$ algorithms is slightly higher at $\Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$ and $\Omega\left(\frac{|\mathcal{X}|^{4\log|\mathcal{X}|/((1-\alpha)\epsilon)}}{\epsilon^5(1-\alpha)\alpha^4}\log\frac{p}{\delta}\right)$ respectively (since $\frac{2\log|\mathcal{X}|}{\epsilon} > \Delta$). However the computation complexity of the LIT algorithm is $O(p^{2\Delta+1}\log p)$ and that of the CVDT algorithm is $O(|\mathcal{X}|^{\Delta}p^{\Delta+2}n)$, which is much larger that $O(\frac{p}{\epsilon}\Delta\xi n)$ for $RecGreedy(\epsilon)$ algorithm, $O(\frac{p}{(1-\alpha)\epsilon}\xi n)$ for the $FbGreedy(\epsilon, \alpha)$ algorithm and $O(\frac{p}{\epsilon}(\xi+1)n)$ for $GreedyP(\epsilon)$ algorithm (since $\xi = O(\log p)$ and $\xi < \Delta^R$ when (A2) holds). Recall however that using the correlation decay property and super-neighborhood selection, the computation complexity of search based algorithms can be decreased. In [11], Bresler et al. showed that by assuming exponential correlation decay (with parameter μ) and a super-neighborhood selection, the LIT algorithm has a run-time of $O(p\Delta^{\frac{\Delta\log(4/\beta)}{\mu}}n)$. We have the following proposition for the CVDT algorithm.

Proposition 2 *Consider a graphical model $M = (G, X)$, where $G = (V, E)$ have maximum degree Δ, satisfying correlation decay (A2). Then by super-neighborhood selection the CVDT algorithm has an expected run-time of $O(p\Delta^{(\Delta+1)R}|\mathcal{X}|^{\Delta}n)$, when the super-neighborhood is chosen as* (7).

However with correlation decay (A2) the run-time of Algorithms 1, 2 and 3 are $O(\frac{p}{\epsilon}\Delta^{R+1}n)$, $O(\frac{p}{(1-\alpha)\epsilon}\Delta^R n)$ and $O(\frac{p}{\epsilon}\Delta^R n)$ respectively still smaller than the LIT and CVDT algorithms.

C. Comparison with convex optimization based algorithms:

In [10] Ravikumar et al. presented a convex optimization based learning algorithm for Ising models, which we have referred as the RWL algorithm. It was later extended for any pairwise graphical model by Jalali et al. in [12]. These algorithms assume extra incoherence or restricted strong convexity conditions hold, in which case they have a low sample complexity of $\Omega(\Delta^3\log p)$. However these algorithms have a computation complexity of $O(p^4)$ higher than the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms. Moreover the greedy algorithms we propose are applicable for a wider class of graphical models. Finally these optimization based algorithms require a strong incoherence property to guarantee its success; conditions which may not hold even for a large class of Ising models as shown by Bento et al. in [18]. They also prove that the RWL algorithm fails in a diamond network (Figure 2) for a large enough degree, whenever there is a strong correlation between non-neighbors, our algorithm successfully recovers the correct graph in such scenarios. In our simulations later we will see that the failure of the RWL algorithm for the diamond network exactly corresponds to the case $\Delta > D_{th} = \frac{2\theta}{\log\cosh(2\theta)} + 1$, which is also when the $Greedy(\epsilon)$ fails. In [18] the authors prove that for a given Δ the RWL algorithm fails when $\theta < \theta_T$ and this critical threshold θ_T behaves like $\frac{1}{\Delta}$. Now if we define

$$\theta_0 = \max\{\theta : \frac{2\theta}{\log\cosh(2\theta)} + 1 \geq \Delta\} \qquad (10)$$

Then from our simulations for all $\theta < \theta_0$ the RWL algorithm fails. Also this θ_0 is almost equal to $\frac{1}{\Delta}$. Hence we make the following conjecture.

Conjecture 1 *The RWL algorithm fails to recover the correct graph in the diamond network exactly when $\theta < \theta_0$. θ_0 given by equation* (10).

In [19] Jalali et al. presented a forward-backward algorithm based on convex optimization for learning *pairwise graphical models* (as opposed to general graphical models in this paper). It has even lower sample complexity of $\Omega(\Delta^2 \log p)$ and works under slightly milder assumptions than the RWL algorithm.

D. Which greedy algorithm should we use?

From the above performance comparison we can say that $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms can be used to find the graph G efficiently when the super-neighborhood size is small ($\xi = O(\log n)$) in discrete graphical models with correlation decay. A natural question to ask then is which among these three greedy algorithms should we use? The answer depends on the particular application. In terms of sample complexity theoretically $FbGreedy(\epsilon, \alpha)$ has a higher sample complexity than $RecGreedy(\epsilon)$ and $GreedyP(\epsilon)$ algorithms. However the difference is not much for a constant α and in our experiments we see all the three greedy algorithms have similar sample complexities (see Section VII). The theoretical guarantee on computation complexity also varies depending on parameters α, Δ and ξ. However from our experiments we see $RecGreedy(\epsilon)$ has much higher run-time than $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms which show similar run-times. We conclude that in practical applications it is better to use $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms than the $RecGreedy(\epsilon)$ algorithm. Now if we know the bound on maximum degree Δ, after running the $Greedy(\epsilon)$ algorithm if the size of the estimated neighborhood set $\widehat{N}(i)$ is considerably higher than Δ this indicates there are large number of non-neighbors. In such cases $GreedyP(\epsilon)$ may take a considerable time to remove these non-neighbors during the node pruning step and $FbGreedy(\epsilon, \alpha)$ algorithm could have removed much of these nodes in earlier iterations, when the size of the conditioning set was still small, resulting in less computation. This calls for the use of $FbGreedy(\epsilon, \alpha)$ algorithm in these cases. Similarly when size of the set $\widehat{N}(i)$ returned by the $Greedy(\epsilon)$ is comparable or slightly greater than Δ it will be more efficient to use the $GreedyP(\epsilon)$ algorithm (for example in the diamond network and grid network as shown in Section VII).

VII. Simulation Results

In this section we present some simulation results characterizing the performance of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms. We compare the performance with the $Greedy(\epsilon)$ algorithm [13] and the logistic regression based RWL algorithm [10] in Ising models. We consider two graphs, a 4×4 square grid (Figure 3) and the diamond network (Figure 2). In each case we consider an Ising model on the graphs. For the 4×4 grid we take the edge weights $\theta \in \{.25, -.25\}$, generated randomly. For the diamond network we take all equal edge weights $\theta = .25$ or $.5$. Independent and identically distributed samples are generated from the models using Gibbs sampling and the algorithms are run with increasing number of samples. We implement the RWL algorithm using ℓ_1- logistic regression solver by Koh et al. [24] and our algorithms using MATLAB. The parameter ϵ for the greedy algorithms and the ℓ_1 regularization parameter λ for the RWL algorithm are chosen through cross validation which gives the least estimation error on a training dataset. From Theorem 3 and 4 we see that increasing α reduces the sample complexity of $FbGreedy(\epsilon, \alpha)$ algorithm but increases its run-time and vice versa. Hence for practical applications α can be chosen to trade-off between sample complexity and run-time to best suit the application requirements. In our experiments α was taken as .9.

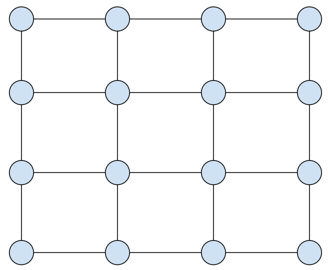
Fig. 3: A 4x4 grid with $\Delta = 4$ and $p = 16$ used for the simulation of the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms.

First we show that for the diamond network (Figure 2) whenever $D > D_{th} = \frac{2\theta}{\log\cosh(2\theta)} + 1$ the RWL algorithm fails to recover the correct graph. We run the RWL algorithm in diamond network with increasing maximum degree D keeping θ fixed. We take $\theta = .25$ for which $D_{th} = \frac{2\times.25}{\log\cosh(2\times.25)} + 1 = 5.16$. The performance is shown in Figure 4. We clearly see that the failure of the RWL algorithm in diamond network corresponds exactly to the case when $D > D_{th}$. The RWL algorithm fails since it predicts a false edge between nodes 0 and $D + 1$. This is surprising since this is also the condition in Proposition 1 which describes the case when $Greedy(\epsilon)$ algorithm fails for the diamond network due to the same reason of estimating a false edge. In some sense $D = D_{th}$ marks the transition between weak and strong correlation between non-neighbors in the diamond network, and both $Greedy(\epsilon)$ and RWL algorithms fail whenever there is a strong correlation. However see next that our greedy Algorithms 1, 2 and 3 succeed even when $D > D_{th}$.

Figure 1 shows the performance of the various algorithms in the case of the diamond network with $p = 6$, $\theta = .5$ and $D = 4 > D_{th} = 3.3$. The $Greedy(\epsilon)$ and RWL algorithms are unable to recover the graph but the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ recover the true graph G, they also show the same error performance. However Figure 5 shows that $GreedyP(\epsilon)$ has a better runtime than the $RecGreedy(\epsilon)$ and $FbGreedy(\epsilon, \alpha)$ algorithms for this diamond network.

Figure 6 shows the performance of the different algorithms for a 4×4 grid network. We see that for this network the RWL algorithm shows a better sample complexity than

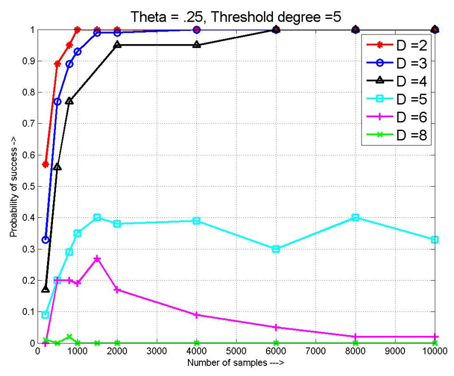

Fig. 4: Performance of the RWL algorithm in diamond network of Figure 2 for varying maximum degree with $\theta = .25$ and $D_{th} = 5$. RWL fails whenever $D > D_{th}$.

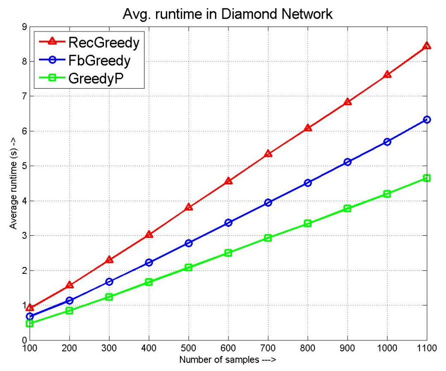

Fig. 5: Figure showing the average runtime performance of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms for the diamond network with $p = 6$, $\Delta = 4$, for varying sample size.

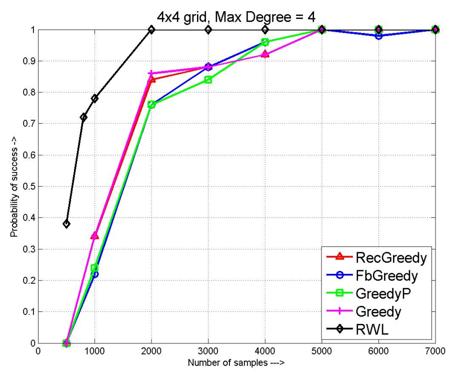

Fig. 6: Performance comparison of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$, $GreedyP(\epsilon)$, $Greedy(\epsilon)$ and RWL algorithms in a 4×4 grid with $p = 16$, $\Delta = 4$ for varying sample size. The error event is defined as $\mathcal{E} = \{\exists i \in V | \widehat{\mathcal{N}_i} \neq \mathcal{N}_i\}$. All three greedy algorithms have the same error performance for this graph.

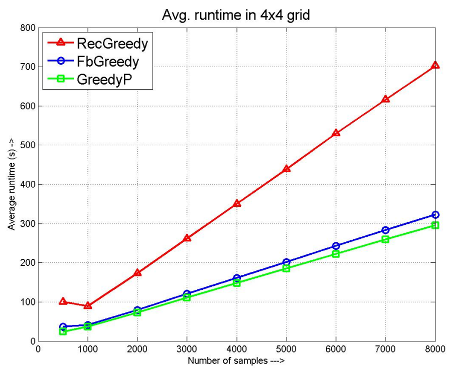

Fig. 7: Figure showing the average runtime performance of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms for the 4×4 grid network with $p = 16$, $\Delta = 4$, with varying sample size.

$RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ or $GreedyP(\epsilon)$ as predicted by the performance analysis. This network exhibits a weak correlation among non-neighbors, hence the $Greedy(\epsilon)$ is able to correctly recover the graph, which obviously implies that the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ also correctly recovers the graph, and all have the same performance.

Figure 7 shows that the $GreedyP(\epsilon)$ algorithm also has the best runtime in the 4×4 grid network among the new greedy algorithms.

References

[1] A. Ray, S. Sanghavi and S. Shakkottai, "Greedy Learning of Graphical Models with Small Girth", Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing, October 2012.

[2] E. Ising, "Beitrag zur theorie der ferromagnetismus", *Zeitschrift fur Physik* 31, pp. 253–258, 1925.

[3] C. D. Manning and H. Schutze, "Foundations of Statistical Natural Language Processing", MIT Press, Cambridge, MA. MR1722790, 1999.

[4] G. Cross and A. Jain, "Markov random field texture models", *IEEE Trans. PAMI*, 5, pp. 25–39, 1983.

[5] M. Hassner and J. Sklansky, "The use of Markov random fields as models of texture", *Comp. Graphics Image Proc.* 12, pp. 357–370, 1980.

[6] S. Wasserman and P. Pattison "Logit models and logistic regressions for social networks 1. An introduction to Markov graphs and $p*$", *Psychometrika* 61, pp. 401-425, 1996.

[7] A. Dobra, C. Hans, B. Jones, J. R. Nevins, G. Yao, and M. West, "Sparse graphical models for exploring gene expression data", *J. Multiv. Anal. 90*, 196–212, 2004.

[8] D. Karger, N. Srebro, "Learning Markov networks: maximum bounded tree-width graphs", *Symposium on Discrete Algorithms*, pp. 392-401, 2001.

[9] C. Chow, C. Liu, "Approximating Discrete Probability Distributions with Dependence Trees", *IEEE Trans. on Information Theory*, vol. 14, pp. 462-467, 1968.

[10] P. Ravikumar, M. Wainwright, J. D. Lafferty, " High-Dimensional Ising Model Selection Using ℓ_1-Regularized Logistic Regression" *The Annals of Statistics*, vol. 38, no. 3, pp. 1287-1319, 2010.

[11] G. Bresler, E. Mossel and A. Sly, " Reconstruction of Markov Random Field from Samples: Some Observations and Algorithms" *Proceedings of the* 11^{th} *international workshop*, APPROX 2008, and *12th international workshop*, RANDOM 2008, pp. 343-356.

[12] A. Jalali, P. Ravikumar, V. Vasuki and S. Sanghavi, "On Learning Discrete Graphical Models using Group-Sparse Regularization", *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[13] P. Netrapalli, S. Banerjee, S. Sanghavi and S. Shakkottai, "Greedy Learn-

ing of Markov Network Structure", 48^{th} *Annual Allerton Conference*, 2010.

[14] A. Anandkumar, V. Y. F Tan and A. S. Willsky, "High-Dimensional Structure Learning of Ising Models: Local Separation Criterion", *http://arxiv.org/abs/1107.1736*, July 2011.

[15] A. Anandkumar, J. E. Yukich, and A. Willsky, "Scaling Laws for Random Spatial Graphical Models", *In Proc. of IEEE ISIT*, Austin, USA, June 2010.

[16] S. Winkler, S. Tatikonda, "Criteria for Rapid Mixing of Gibbs Samplers and Uniqueness of Gibbs Measures", *Allerton Conf. on Communication, Control, and Computing*, 2006.

[17] R. L. Dobrushin, "The Problem of Uniqueness of a Gibbsian Random Field and the Problem of Phase Transitions", *Functional Analysis and its Applications*, vol. 2, pp. 302312, 1968.

[18] J. Bento and A. Montanari, "On the trade-off between complexity and correlation decay in structural learning algorithms", *http://arxiv.org/abs/1110.1769*, 2011.

[19] A. Jalali, C. Johnson, P. Ravikumar, "On Learning Discrete Graphical Models using Greedy Methods", *In Advances in Neural Information Processing Systems (NIPS) 24*, 2011.

[20] D. Weitz, "Counting independent sets up to the tree threshold", *in Proc. of ACM symp. on Theory of Computing*, pp. 140 – 149, 2006.

[21] A. Montanari, "Lecture Notes: Inference in Graphical Models", http://www.stanford.edu/~montanar/TEACHING/Stat375/stat375.html, 2011.

[22] R. Griffiths, "Correlations in Ising ferromagnets", *Journal of Mathematical Physics*, Vol. 8, 478, 1967.

[23] T. Cover and J. Thomas, "Elements of Information Theory". *John Wiley and Sons, Inc.*, 2006.

[24] http://www.stanford.edu/~boyd/l1_logreg/.

APPENDIX

In this section we present the proofs of Lemma 2, 3, 4, 8, Theorem 1, 2, 3, 4 and Proposition 1, 2.

Lemma 2

Proof: Using Azuma's inequality we get for any x_i, x_j

$$\begin{aligned} P(|\widehat{P}(x_i,x_j) - P(x_i,x_j)| > \gamma_1) &\leq 2\exp(-2\gamma_1^2 n) \\ &\leq \frac{\delta_1}{|\mathcal{X}|^2 p^2} \text{ (say)} \end{aligned}$$

Taking union bound over all $x_i, x_j \in \mathcal{X}$ and $i, j \in V$ we get with probability at least $1-\delta_1$

$$|\widehat{P}(x_i,x_j) - P(x_i,x_j)| < \gamma_1 \ \forall x_i, x_j$$

Taking $\gamma_1 = \frac{\beta P_{min}}{8|\mathcal{X}|^2}$ we get with probability at least $1-\delta_1$

$$\begin{aligned} |\widehat{P}(x_i|x_j) - P(x_i|x_j)| &\leq \frac{|\widehat{P}(x_i,x_j) - P(x_i,x_j)|}{P(x_j)} \\ &\quad + \frac{|\widehat{P}(x_j) - P(x_j)|}{P(x_j)} \\ &\leq \frac{\gamma_1}{P_{min}} + \frac{|\mathcal{X}|\gamma_1}{P_{min}} \leq \frac{2|\mathcal{X}|\gamma_1}{P_{min}} \\ &= \frac{\beta}{4|\mathcal{X}|} \end{aligned}$$

■

Lemma 3

Proof: From Lemma 2 it is clear that if number of samples $n = O(\log \frac{p}{\delta_1})$ then with probability at least $1-\delta_1$ equation (9) holds for all $i, j \in V$. Then for any $i \in V, j \in \mathcal{N}_i$ with probability at least $1-\delta_1$ we have

$$\begin{aligned} \hat{\phi}_i(j) &= \max_{x_j, x_j'} ||\widehat{P}(X_i|X_j = x_j) - \widehat{P}(X_i|X_j = x_j')||_{TV} \\ &= \max_{x_j, x_j'} \frac{1}{2} \sum_{x_i \in \mathcal{X}} |\widehat{P}(x_i|x_j) - \widehat{P}(x_i|x_j')| \\ &\geq \beta - \frac{|\mathcal{X}|}{2} \frac{\beta}{2|\mathcal{X}|} \\ &> \frac{\beta}{2} \end{aligned}$$

Equation (7) implies node j is included in the super-neighborhood S_i. Hence for all $i \in V$, $\mathcal{N}_i \subseteq S_i$ with probability greater than $1-\delta_1$.

■

Lemma 4

Proof: Let $\mu = \frac{\beta}{2} - f(R+1)$. From Lemma 2 we know that for $n > \frac{32|\mathcal{X}|^4}{\mu^2 P_{min}^2}\left[2\log|\mathcal{X}|p + \log\frac{2}{\delta_2}\right]$ we have

$$|P(X_i|X_j) - \widehat{P}(X_i|X_j)| < \frac{\mu}{4|\mathcal{X}|}$$

for all $i, j \in V$.

Then for any $j \in V$ such that $d(i,j) \geq R+1$ we have,

$$\begin{aligned} \hat{\phi}_i(j) &= \max_{x_j, x_j'} ||\widehat{P}(X_i|X_j = x_j) - \widehat{P}(X_i|X_j = x_j')||_{TV} \\ &= \max_{x_j, x_j'} \frac{1}{2} \sum_{x_i \in \mathcal{X}} |\widehat{P}(x_i|x_j) - \widehat{P}(x_i|x_j')| \\ &\leq \phi_i(j) + 2\frac{|\mathcal{X}|}{2} \frac{\mu}{4|\mathcal{X}|} \\ &< f(R+1) + \frac{\mu}{4} < \frac{\beta}{2} \end{aligned}$$

Hence with the number of samples $n = O(\log \frac{p}{\delta_2})$ with probability greater than $1-\delta_2$ all nodes $j \in V$ such that $d(i,j) \geq R+1$ are not included in S_i. Hence $|S_i| < \Delta_i \Delta^{R-1}$ with probability greater than $1-\delta_2$.

■

Theorem 1

Proof: From Lemma 1 we know that for any graphical model satisfying Dobrushin's condition the correlation decay function is exponential, $f(x) = \frac{\gamma^x}{1-\gamma}$. Now from Lemma 4 we know that with $n = \Omega(\log \frac{p}{\delta_4})$ samples the super-neighborhood size is bounded by $\xi < \Delta^R$ with high probability. For exponential correlation decay the super-neighborhood radius can be bounded by $R < \log \frac{2}{(1-\gamma)\beta} / \log \frac{1}{\gamma}$. The theorem follows.

■

Theorem 2

Proof: It is easy to show that in a zero field Ising model with maximum degree Δ and edge weights θ the Dobrushin's condition is satisfied for $\gamma = \Delta \tanh 2\theta < 1$. Hence such Ising model will show exponential correlation decay and from Theorem 1 with $n = \Omega(\log \frac{p}{\delta_5})$ samples with high probability the super-neighborhood size will be bounded by $\xi < \Delta^{\log \frac{2}{(1-\gamma)\beta} / \log \frac{1}{\gamma}}$. Now as shown in [18] the correlation between neighbors X_i and X_j can be bounded by $C_{ij} = E[X_i X_j] \geq \tanh\theta$. However from symmetry of the partition function as argued in [22] we have $P(X_i = 1, X_j = -1) = P(X_i = -1, X_j = 1)$ and $P(X_i = 1, X_j = 1) = P(X_i = -1, X_j = -1)$. Hence it can be shown that even $\phi_i(j) \geq \tanh\theta$. Now taking $\gamma = \Delta \tanh 2\theta$ and $\beta = \tanh\theta$ in Theorem 1 gives the result. ■

Lemma 8

Proof: The proof is similar to that in [13]. Let $S \subset V$ such that $|S| \leq s$. For any $i \in V$ using Azuma's inequality we get,

$$\begin{aligned} P(|\widehat{P}(x_i, x_S) - P(x_i, x_S)| > \gamma_3) &\leq 2\exp(-2\gamma_3^2 n) \\ &\leq \frac{2\delta_3}{(2|\mathcal{X}|p)^{s+1}} \text{ (say)} \end{aligned}$$

Now taking union bound over all $i \in V$, $S \subset V$, $|S| \leq s$ and all $x_i \in \mathcal{X}$, $x_S \in \mathcal{X}^{|S|}$, with probability at least $1 - \delta_3$ we have $|\widehat{P}(x_i, x_S) - P(x_i, x_S)| < \gamma_3$, for any $i \in V$, $S \subset V$, $|S| \leq s$. This implies

$$\begin{aligned} ||\widehat{P}(X_i, X_S) - P(X_i, X_S)||_{TV} &\leq \frac{|\mathcal{X}|^{(s+1)}}{2}\gamma_3 \\ ||\widehat{P}(X_S) - P(X_S)||_{TV} &\leq \frac{|\mathcal{X}|^{(s+1)}}{2}\gamma_3 \end{aligned}$$

Now taking $\gamma_3 = \frac{\epsilon^2\alpha^2}{256|\mathcal{X}|^{s+2}}$ and using Lemma 5 we get,

$$\begin{aligned} |\widehat{H}(X_i|X_S) - H(X_i|X_S)| &\leq |\widehat{H}(X_i, X_S) - H(X_i, X_S)| + \\ &\quad |\widehat{H}(X_S) - H(X_S)| \\ &\leq |\mathcal{X}| \left(\frac{2||\widehat{P}(X_i, X_S) - P(X_i, X_S)||_{TV}}{|\mathcal{X}|} \right. \\ &\quad \log \frac{|\mathcal{X}|}{2||\widehat{P}(X_i, X_S) - P(X_i, X_S)||_{TV}} + \\ &\quad \left. \frac{2||\widehat{P}(X_S) - P(X_S)||_{TV}}{|\mathcal{X}|} \log \frac{|\mathcal{X}|}{2||\widehat{P}(X_S) - P(X_S)||_{TV}} \right) \\ &\leq 2|\mathcal{X}|\sqrt{|\mathcal{X}|^s\gamma_3} < \frac{\alpha\epsilon}{8} = \zeta \end{aligned}$$

■

Theorem 3

Proof: The proof of correctness when $P(X)$ is known is straight forward. From local Markov property (1) the conditional entropy $H(X_i|X_{\mathcal{N}_i}) = H(X_i|X_V) < H(X_i|X_A)$ for any set A not containing all the neighbors $\mathcal{N}_i$. From degeneracy assumption (A1) including a neighboring node in the conditioning set always produce a decrease in entropy by at least ϵ. In $RecGreedy(\epsilon)$ in each iteration the algorithm runs till all the neighbors $\mathcal{N}_i$ are included in the conditioning set and the last added node is always a neighbor. In $GreedyP(\epsilon)$ nodes are added till all neighbors have been included in the conditioning set. Then in the pruning step removing a non-neighbor does not increase the entropy, therefore all spurious nodes are detected and removed. In $FbGreedy(\epsilon, \alpha)$ each iteration decrease entropy by at least $(1-\alpha)\epsilon/2$. Since the entropy is bounded it terminates in a finite number of steps and minimum is reached only when all neighbors have been added to the conditioning set. All spurious nodes get eliminated by the backward steps (in earlier iterations or after all neighbors are added).

Now we give the proof of sample complexity when we have samples. Define the error event $\mathcal{E} = \{\exists S \subset V, |S| < \xi|\ |\widehat{H}(X_i|X_S) - H(X_i|X_S)| > \frac{\epsilon}{8}\}$. Note that when $\mathcal{E}^c$ occurs we have for any $i \in V$, $j \in \mathcal{N}_i$, $A \subset V\backslash\{i,j\}$, $|A| < \xi$

$$\begin{aligned} \widehat{H}(X_i|X_A) - \widehat{H}(X_i|X_A, X_j) &\geq H(X_i|X_A) \\ -H(X_i|X_A, X_j) - \frac{\epsilon}{4} &> \frac{3\epsilon}{4} \end{aligned} \tag{11}$$

which follows from equation (3).

Proof for RecGreedy(ϵ) algorithm: We first show that when $\mathcal{E}^c$ occurs the $RecGreedy(\epsilon)$ correctly estimates the graph G. The proof is by induction. Let $\mathcal{N}_i = \{j_1, \ldots, j_{\Delta_i}\} \subset S_i$ since super-neighborhoods S_i are given. Let r denote the counter indicating the number of times the outermost while loop has run and s be the counter indicating the number of times the inner while loop has run in a particular iteration of the outer while loop. Clearly $s \leq |S_i|$. In the first step since $\widehat{T}(i) = \phi$ the algorithm finds the node $k \in S_i$ such that $\widehat{H}(X_i|X_k)$ is minimized and adds it to $\widehat{T}(i)$. Suppose it runs till $s = s_1$ such that $\mathcal{N}_i \not\subset \widehat{T}(i)$, then $\exists$ some $j_l \in \mathcal{N}_i$ such that $j_l \notin \widehat{T}(i)$. Then from equation (11) $\widehat{H}(X_i|X_{\widehat{T}(i)}, X_{j_l}) < \widehat{H}(X_i|X_{\widehat{T}(i)}) - \epsilon/2$. Hence $\min_{k \in S_i - \widehat{T}(i)} \widehat{H}(X_i|X_{\widehat{T}(i)}, X_k) < \widehat{H}(X_i|X_{\widehat{T}(i)}) - \epsilon/2$. Therefore the control goes to the next iteration $s = s_1 + 1$. However after the last neighbor say j_l is added to $\widehat{T}(i)$ we have

$$\begin{aligned} &|\widehat{H}(X_i|X_{\widehat{T}(i)}, X_k) - \widehat{H}(X_i|X_{\widehat{T}(i)})| \\ \leq\ &|H(X_i|X_{\widehat{T}(i)}, X_k) - H(X_i|X_{\widehat{T}(i)})| + \frac{\epsilon}{4} \\ =\ &0 + \frac{\epsilon}{4} = \frac{\epsilon}{4} < \frac{\epsilon}{2} \end{aligned} \tag{12}$$

for any $k \in S_i - \widehat{T}(i)$. Thus j_l is added to $\widehat{N}_i$, variable *complete* is set to TRUE and the control exits the inner while loop going to the next iteration $r = r+1$. Proceeding similarly one neighboring node is discovered in each iteration $r = 1$ to $r = \Delta_i$. At $r = \Delta_i + 1$, $\widehat{N}(i) = \mathcal{N}_i$. Thus in step $s = 1$, $\widehat{T}(i) = \mathcal{N}_i$, so the entropy cannot be reduced further. Hence variable $iterate$ is set to FALSE and control exits the outer while loop returning the correct neighborhood $\widehat{N}(i) = \mathcal{N}_i$. Lemma 6 bounds the number of steps in each iteration by $\frac{2\log|\mathcal{X}|}{\epsilon}$.

Now taking $\delta_3 = \delta$, $s = \frac{2\log|\mathcal{X}|}{\epsilon}$, $\zeta = \frac{\epsilon}{8}$ in Lemma 8 we have for $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$, $P(\mathcal{E}) \leq \delta$.

Therefore with probability greater than $1 - \delta$ the $RecGreedy(\epsilon)$ correctly recovers G.

Proof for FbGreedy(ϵ, α) algorithm: Define $\mathcal{E} = \{\exists S \subset V, |S| < \xi|\ |\widehat{H}(X_i|X_S) - H(X_i|X_S)| > \frac{\alpha\epsilon}{8}\}$. Let s denote the number of iterations of the while loop. When $\mathcal{E}^c$ occurs we have for any $i \in V$, $j \in \mathcal{N}_i$, $A \subset V\backslash\{i,j\}$, $|A| < \xi$

$$\begin{aligned}\widehat{H}(X_i|X_A) - \widehat{H}(X_i|X_A, X_j) &\geq H(X_i|X_A) \\ -H(X_i|X_A, X_j) - \frac{\alpha\epsilon}{4} &> \frac{3\epsilon}{4} \qquad (13)\end{aligned}$$

Again we prove by induction. For $s = 1$ the forward step adds a node to the conditioning set $\widehat{N}(i)$ as shown previously for the $RecGreedy(\epsilon)$ algorithm. Consider iteration $s > 1$. Note that it is enough to show the following.

- In each iteration the backward step never removes a neighboring node $j \in \mathcal{N}_i$.
- After the last neighbor is added to the conditioning set $\widehat{N}(i)$ the backward step removes all non-neighbors if any.

From equation (13) it is clear that removing a neighboring node $j \in \widehat{N}(i) \bigcap \mathcal{N}_i$ increases the entropy by at least $\frac{3\epsilon}{4} > \frac{\alpha\epsilon}{2}$. Hence a neighboring node is never removed in the backward step. If there exists a non-neighbor $l \in \widehat{N}(i)$ such that $\widehat{H}(X_i|X_{\widehat{N}(i)\setminus l}) - \widehat{H}(X_i|X_{\widehat{N}(i)}) < \frac{\alpha\epsilon}{2}$ and it produces the least increase in entropy then it gets removed from $\widehat{N}(i)$ and we go to iteration $s+1$. This continues till the forward step had added all neighbors $j \in \mathcal{N}_i$ to the conditioning set. After adding the last neighbor to the conditioning set equation (12) ensures that the forward step adds no other nodes to the conditioning set $\widehat{N}(i)$. If $\widehat{N}(i) = \mathcal{N}_i$ we are done. Else for any non-neighbor $j \in \widehat{N}(i)$ we have,

$$\begin{aligned}&|\widehat{H}(X_i|X_{\widehat{N}(i)\setminus j}) - \widehat{H}(X_i|X_{\widehat{N}(i)})| \\ \leq\ & |H(X_i|X_{\widehat{N}(i)\setminus j}) - H(X_i|X_{\widehat{N}(i)})| + \frac{\alpha\epsilon}{4} \\ =\ & 0 + \frac{\alpha\epsilon}{4} = \frac{\alpha\epsilon}{4} < \frac{\alpha\epsilon}{2} \qquad (14)\end{aligned}$$

Hence the backward step will remove j from the conditioning set (or any other non-neighbor that produces the least increase in entropy). This occurs till all non-neighbors are removed and $\widehat{N}(i) = \mathcal{N}_i$ when neither the forward or the backward step works. The flag complete is then set to TRUE and Algorithm 2 exits the while loop giving the correct neighborhood of node i. Again from Lemma 7 the number of steps required for convergence is bounded by $\frac{4\log|\mathcal{X}|}{(1-\alpha)\epsilon}$. As shown previously for the $RecGreedy(\epsilon)$ algorithm from Lemma 8 with $\delta_3 = \delta$, $s = \frac{4\log|\mathcal{X}|}{(1-\alpha)\epsilon}$ and $\zeta = \frac{\alpha\epsilon}{8}$ for $n = \Omega\left(\frac{|\mathcal{X}|^{4\log|\mathcal{X}|/((1-\alpha)\epsilon)}}{(1-\alpha)\alpha^4\epsilon^5}\log\frac{p}{\delta}\right)$ the probability of error $P(\mathcal{E}) \leq \delta$. Therefore the $FbGreedy(\epsilon, \alpha)$ succeeds with probability at least $1-\delta$. This completes the proof.

Proof for GreedyP(ϵ) algorithm: The proof is similar to that for the $RecGreedy(\epsilon)$ algorithm. Let event $\mathcal{E}$ be as defined in the proof for $RecGreedy(\epsilon)$ algorithm. When event $\mathcal{E}^c$ occurs the $Greedy(\epsilon)$ runs till all neighbors are added to the set $\widehat{N}(i)$. Then for non-neighbors $j \in \widehat{N}(i)$

$$\begin{aligned}&|\widehat{H}(X_i|X_{\widehat{N}(i)\setminus j}) - \widehat{H}(X_i|X_{\widehat{N}(i)})| \\ \leq\ & |H(X_i|X_{\widehat{N}(i)\setminus j}) - H(X_i|X_{\widehat{N}(i)})| + \frac{\epsilon}{4} \\ =\ & 0 + \frac{\epsilon}{4} = \frac{\epsilon}{4} < \frac{\epsilon}{2}\end{aligned}$$

Hence j is removed from $\widehat{N}(i)$. But for any neighbor $k \in \mathcal{N}(i)$

$$\begin{aligned}&|\widehat{H}(X_i|X_{\widehat{N}(i)\setminus k}) - \widehat{H}(X_i|X_{\widehat{N}(i)})| \\ \geq\ & |H(X_i|X_{\widehat{N}(i)\setminus j}) - H(X_i|X_{\widehat{N}(i)})| - \frac{\epsilon}{4} \\ \geq\ & \epsilon - \frac{\epsilon}{4} = \frac{3\epsilon}{4} > \frac{\epsilon}{2}\end{aligned}$$

Thus the neighbors are not eliminated. The algorithm terminates after all non-neighbors have been eliminated. The probability of error is upper bounded by $P(\mathcal{E}) \leq \delta$ with number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$. ■

Theorem 4

Proof: First consider the $RecGreedy(\epsilon)$ algorithm. With probability $1-\delta$ Algorithm 1 finds the correct neighborhood of each node i. In this case from Lemma 6 the number of steps in each recursion is $O(\frac{1}{\epsilon})$, the search in each step takes $O(\xi)$ time, number of recursions is at most Δ and the entropy calculation takes $O(n)$ time for each node i. Hence the overall runtime is $O(\frac{p}{\epsilon}\Delta\xi n)$. When the algorithm makes an error with probability δ the number of steps and the number of recursions are bounded by $O(\xi)$. Hence the overall expected runtime is $O(\delta p\xi^3 n + (1-\delta)\frac{p}{\epsilon}\Delta\xi n)$.

For the $FbGreedy(\epsilon, \alpha)$ algorithm from Lemma 7 we know that the number of steps is $O(\frac{1}{(1-\alpha)\epsilon})$. The search in either the forward or backward step is bounded by ξ and the entropy calculation takes $O(n)$ time. Hence when the algorithm succeeds the run time is $O(\frac{p}{(1-\alpha)\epsilon}\xi n)$. Note that even when the algorithm fails with probability δ, we can prevent going into infinite loops by making sure that once the forward step stopped it is never restarted. Hence the number of steps will still be $O(\frac{1}{(1-\alpha)\epsilon})$ and the overall runtime remains the same. Thus the expected runtime is $O(\frac{p}{(1-\alpha)\epsilon}\xi n)$.

In $GreedyP(\epsilon)$ algorithm when it succeeds with probability $1-\delta$, for each node $i \in V$, the $Greedy(\epsilon)$ takes at most $\frac{2\log|\mathcal{X}|}{\epsilon}$ steps. In each step search set is bounded by ξ and conditional entropy computation takes $O(n)$ time. After greedy algorithm terminates $|\widehat{N}(i)| \leq \frac{2\log|\mathcal{X}|}{\epsilon}$ since one node has been added in each step. Hence number iterations in pruning step is bounded by $\frac{2\log|\mathcal{X}|}{\epsilon}$ and again conditional entropy computation take $O(n)$ time. Hence the total run-time is $O(\frac{p}{\epsilon}\xi n + \frac{p}{\epsilon}n) = O(\frac{p}{\epsilon}(\xi+1)n)$. When error occurs the number of greedy steps and pruning iterations is bounded by ξ. Therefor the expected run-time is $O(\delta p\xi(\xi+1)n + (1-\delta)\frac{p}{\epsilon}(\xi+1)n)$. ■

Proposition 1

Proof: Define $H(a) = a\log(\frac{1}{a}) + (1-a)\log(\frac{1}{1-a})$ for $0 \leq a \leq 1$. Then simple calculation shows $H(X_0|X_{D+1}) = H(p)$ and $H(X_0|X_1) = H(q)$ where

$$\begin{aligned}p &= \frac{2^{D+1}}{2^{D+1} + 2(e^{2\theta} + e^{-2\theta})^D} \\ q &= \frac{2^D + 2e^{-2\theta}(e^{2\theta} + e^{-2\theta})^{D-1}}{2^{D+1} + 2(e^{2\theta} + e^{-2\theta})^D}\end{aligned}$$

Note that $p < \frac{1}{2}$ and $q < \frac{1}{2}$. Since $H(a)$ is monotonic increasing for $0 < a < \frac{1}{2}$, $H(X_0|X_{D+1}) < H(X_0|X_1)$ iff $p < q$. This implies

$$\begin{aligned}
2^{D+1} &< 2^D + 2e^{-2\theta}(e^{2\theta} + e^{-2\theta})^{D-1} \\
2 &< 1 + e^{-2\theta}\left(\frac{e^{2\theta} + e^{-2\theta}}{2}\right)^{D-1} \\
e^{2\theta} &< \left(\frac{e^{2\theta} + e^{-2\theta}}{2}\right)^{D-1} \\
D &> \frac{2\theta}{\log\left(\frac{e^{2\theta}+e^{-2\theta}}{2}\right)} + 1 = \frac{2\theta}{\log\cosh(2\theta)} + 1
\end{aligned}$$

■

Proposition 2

Proof: For each node $i \in V$ its distribution can be made conditionally independent of all other nodes except the neighbors. In order to find $\mathcal{N}(i)$ the CVDT algorithm maybe run within the super-neighborhood set S_i to reduce its computation complexity. The minimum size of the separator is upper bounded Δ. Let $|S_i| < \Delta^R$. Then CVDT searches over all possible conditioning set of size Δ, which takes $\binom{\Delta^R}{\Delta} = O(\Delta^{\Delta R})$ iterations. For each conditioning set it takes $O(n)$ time to compute the conditional variation distance and $|\mathcal{X}|^\Delta$ iterations to find the maximum conditional variation distance. Therefore the expected runtime is $O(p\Delta^{(\Delta+1)R}|\mathcal{X}|^\Delta n)$.

■